Importing required Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# read the file and display some of the data to check
df = pd.read_csv('auto-mpg.csv')
df.head(5)
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

In [3]:
```
# Understanding the dimensions of the dataset
df.shape
```

Out[3]:
```
(398, 9)
```

In [4]:
```
# Exploring the dataset by datatypes and null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
# Knowing the unique values in features to know it is categorical or continous
df.nunique()
```

Out[7]:
```
mpg             129
cylinders         5
displacement     82
horsepower       94
weight          351
acceleration     95
model year       13
origin            3
car name        305
dtype: int64
```

In [8]:
```
# Check for wrong inputs
temp  = pd.DataFrame(df.horsepower.str.isdigit())
temp[temp['horsepower']==False]
```

Out[8]:

	horsepower
32	False
126	False
330	False
336	False
354	False
374	False

- As these won tbe shown in missing values but we need to clear this
- Because the horsepower column has numerical values but its shown as object type
- To further evaluate we nee to change it to numerical datatype.

In [9]:
```
#Finding the str value
rows = [32,126,330,336,354,374]
columns = 'horsepower'
df.loc[rows,columns]
```

Out[9]:
```
32     ?
126    ?
330    ?
336    ?
354    ?
374    ?
Name: horsepower, dtype: object
```

In [10]:
```
df['horsepower'] = pd.to_numeric(df.horsepower.str.replace('?','NAN'),errors='coerce')
```

In [11]:
```
df['horsepower'].isnull().sum()
```

Out[11]:
```
6
```

In [12]:
```
df['horsepower'] = df['horsepower'].fillna(df['horsepower'].median())
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64
 7   origin        398 non-null    int64
 8   car name      398 non-null    object
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

In [15]:
```
df.describe().T
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6
cylinders	398.0	5.454774	1.701004	3.0	4.000	4.0	8.000	8.0
displacement	398.0	193.425879	104.269838	68.0	104.250	148.5	262.000	455.0
horsepower	398.0	104.304020	38.222625	46.0	76.000	93.5	125.000	230.0
weight	398.0	2970.424623	846.841774	1613.0	2223.750	2803.5	3608.000	5140.0
acceleration	398.0	15.568090	2.757689	8.0	13.825	15.5	17.175	24.8
model year	398.0	76.010050	3.697627	70.0	73.000	76.0	79.000	82.0
origin	398.0	1.572864	0.802055	1.0	1.000	1.0	2.000	3.0

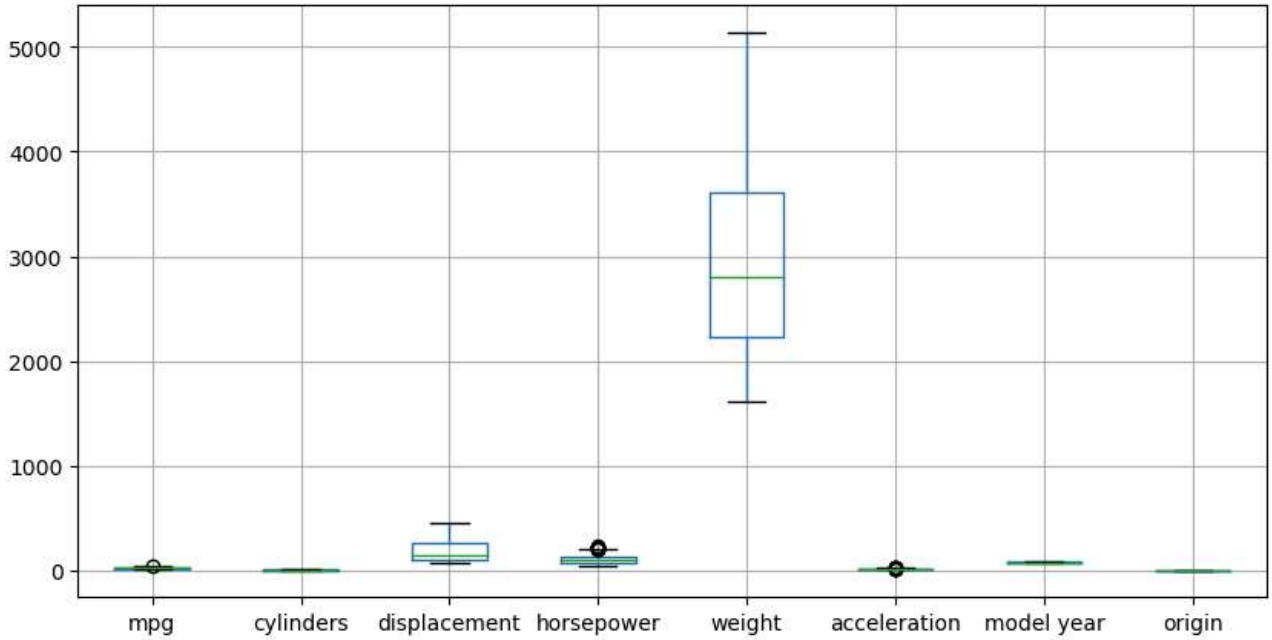
In [16]:
```
df['car name'].nunique()
```

Out[16]:
```
305
```

- We are going to drop the car name column
- It's not essential for analysis
- Cant encode or modify to significant variable
- Because its continous str variable.

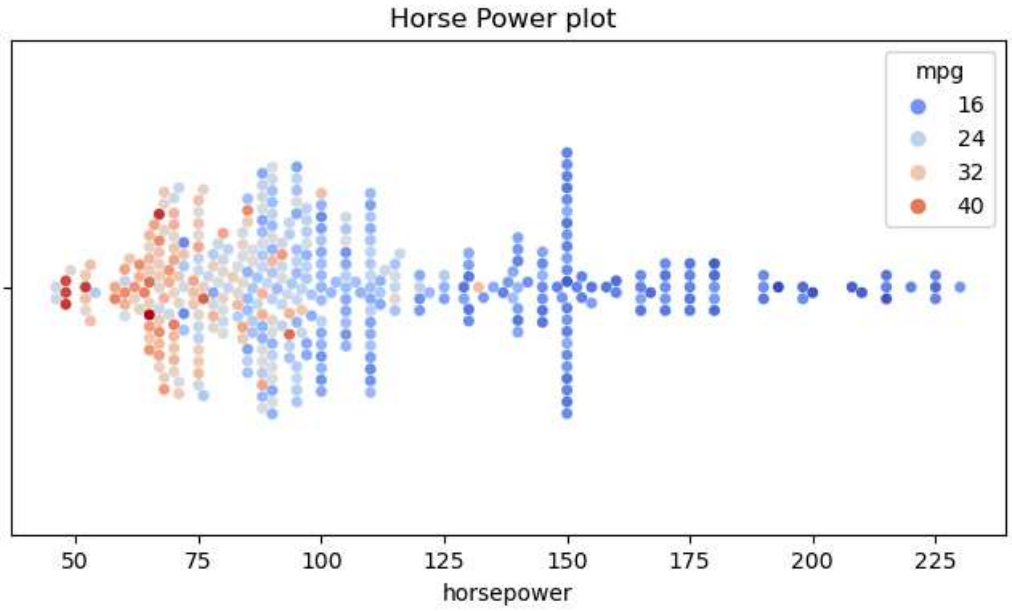
In [17]:
```
df.drop(columns=['car name'],axis=1,inplace=True)
```

In [18]:
```
# Find the outliers and need to treat them if necessary
plt.figure(figsize=(10,5))
df.boxplot()
plt.show()
```

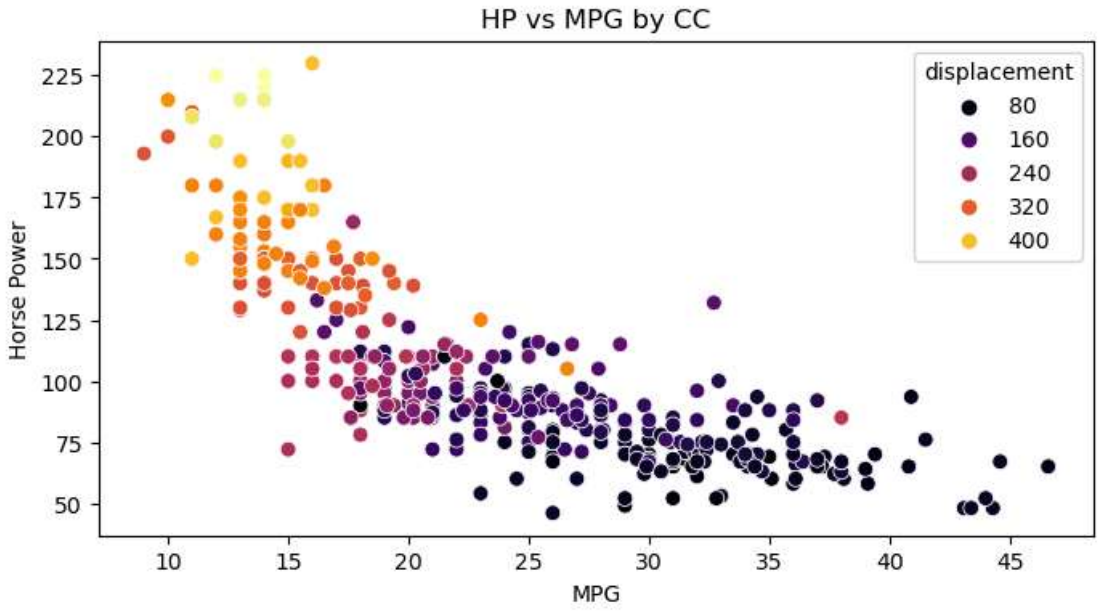


Visualization

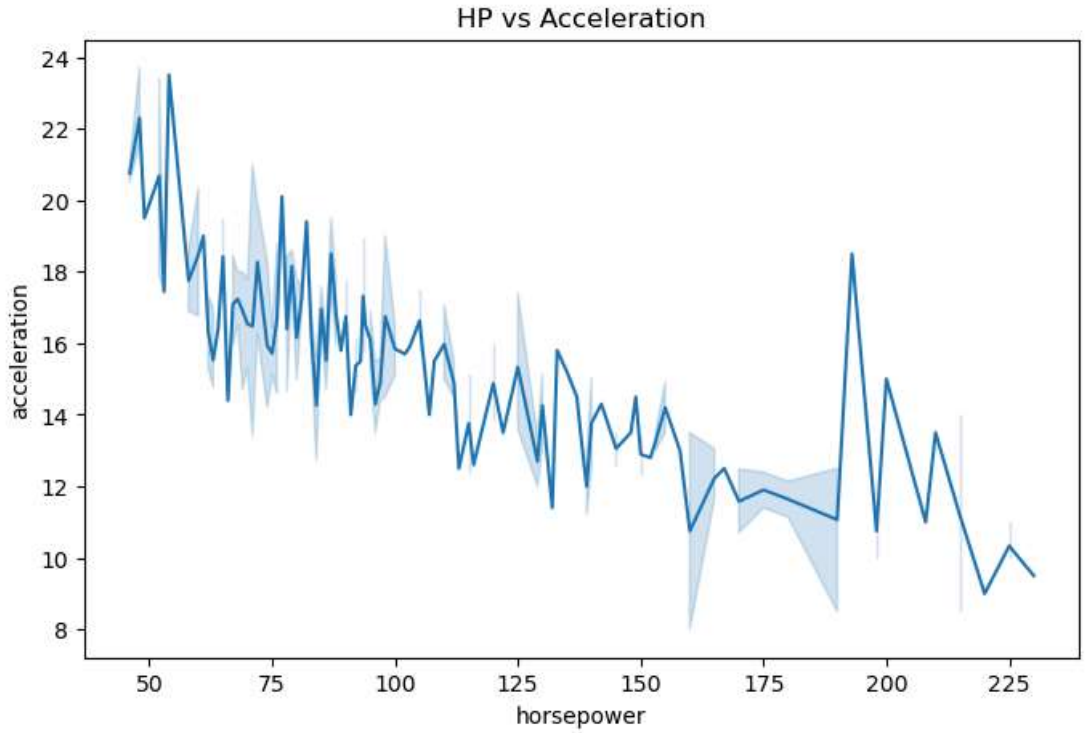
In [19]:
```
# Visualing horsepower to mpg
plt.figure(figsize=(8,4))
sns.swarmplot(x=df['horsepower'],hue=df['mpg'],palette='coolwarm')
plt.title('Horse Power plot ')
plt.show()
```



In [20]:
```
plt.figure(figsize=(8,4))
sns.scatterplot(hue='displacement', y='horsepower', data=df, x='mpg', palette='inferno', s=50)
plt.title('HP vs MPG by CC')
plt.xlabel('MPG')
plt.ylabel('Horse Power')
plt.show()
```



In [21]:
```
plt.figure(figsize=(8, 5))
sns.lineplot(x='horsepower', y='acceleration', data=df)
plt.title('HP vs Acceleration')
plt.show()
```



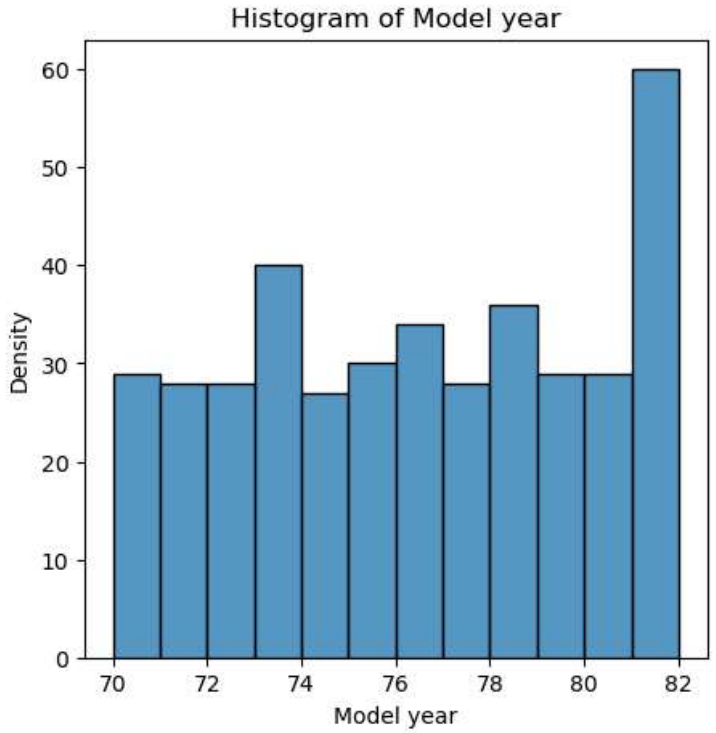
Through visualizations above you can say.

The relationship betweeen the features.

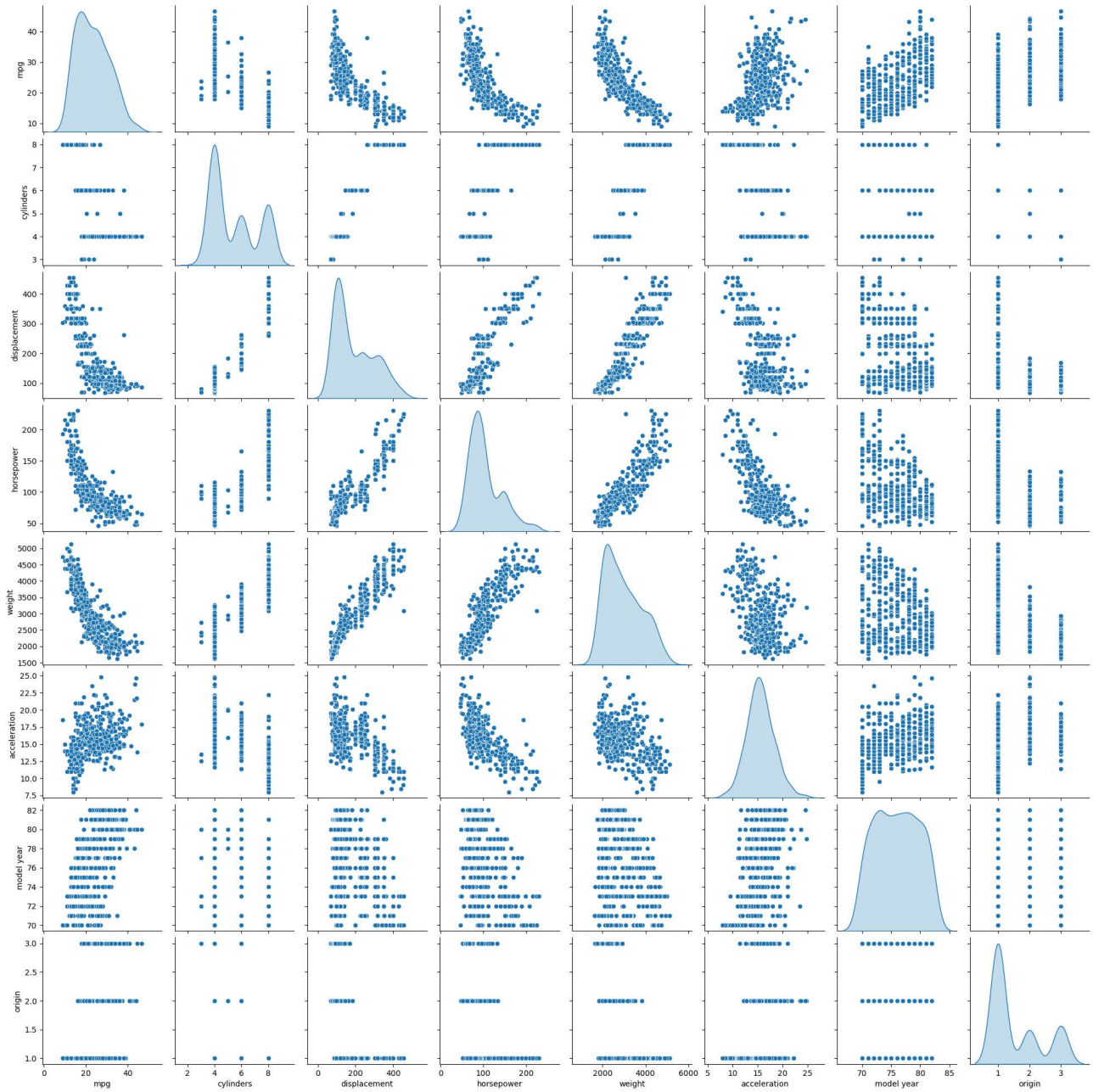
Directly or inversly proportions are been found.

In [22]:
```
# cars of different Model year
plt.figure(figsize=(5, 5))
sns.histplot(data=df, x='model year',bins=12)
plt.xlabel('Model year')
plt.ylabel('Density')
```

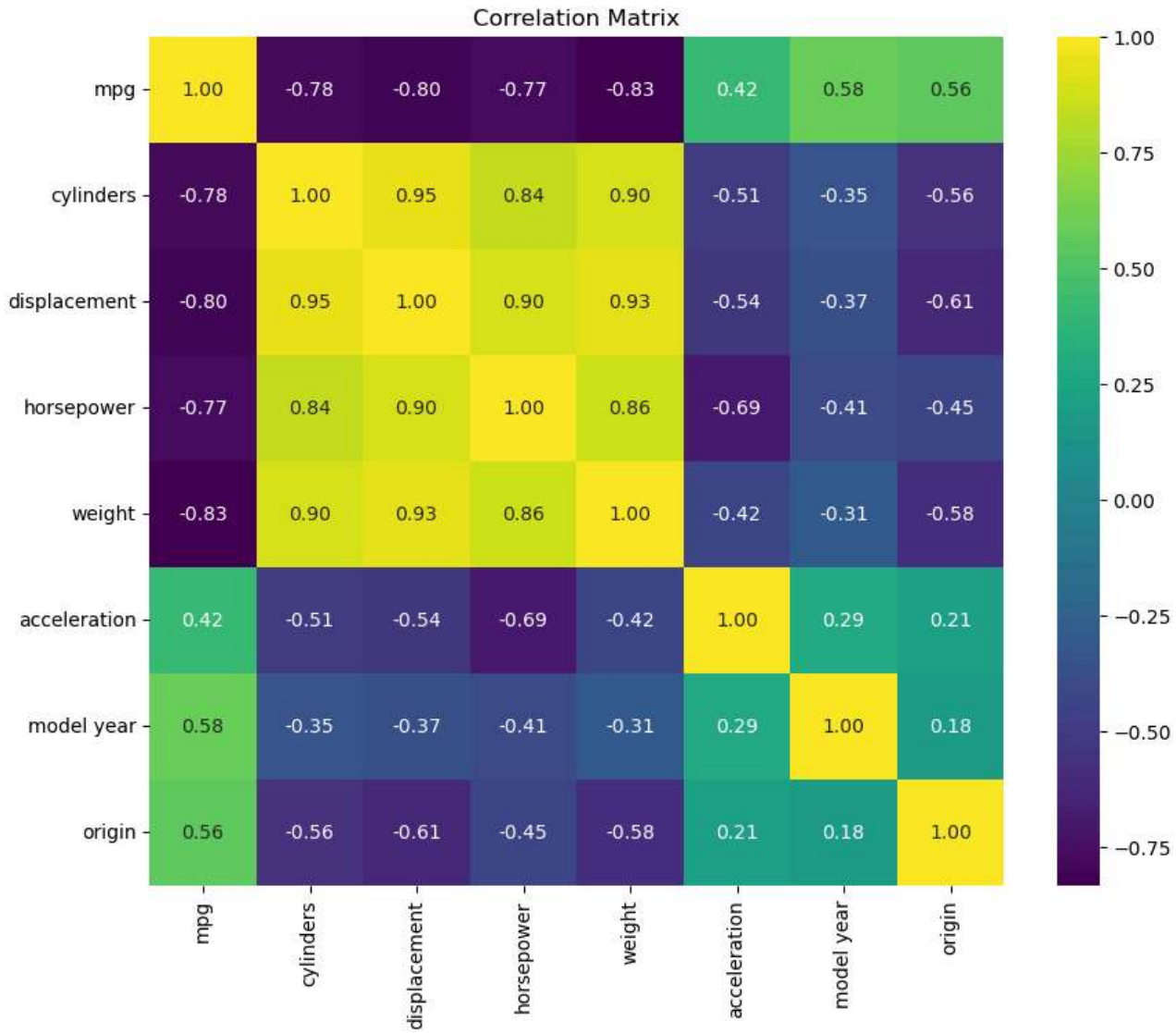
```
plt.title('Histogram of Model year')
plt.show()
```



In [23]:
```
# Pair plot shows the relationship and Multi-variant plots
sns.pairplot(df,diag_kind='kde')
plt.savefig('pairplot.png',dpi=300)
plt.show()
```



In [24]:
```
# Co-relation matrix shows the relationship of variables
corr = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='viridis', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```



Weight, displacement, cyclinder and horsepower has high multi colinearity

We can see the Multi colinearity between the dependent variables also.

Train & Test - Split

In [25]:
```
# importing libraries and doing the train and test split
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = df.drop('mpg',axis=1)
y = df[['mpg']]

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size= 0.30,random_state = 1)
```

In [26]:
```
#printing the shape of these splits to verify
print('X train shape: ',X_train.shape)
print('X test shape: ',X_test.shape)
print('y train shape: ',y_train.shape)
print('y test shape: ',y_test.shape)
```

```
X train shape:  (278, 7)
X test shape:  (120, 7)
y train shape:  (278, 1)
y test shape:  (120, 1)
```

In [27]:
```
# Scaling the data
from sklearn.preprocessing import StandardScaler


ss=StandardScaler()


X_train.iloc[:,:] = ss.fit_transform(X_train.iloc[:,:])

X_test.iloc[:,:] = ss.transform(X_test.iloc[:,:])
```

In [28]:
```
# Fitting the model
lr = LinearRegression()

lin_reg = lr.fit(X_train,y_train)
```

In [29]:
```
# Let us explore the coefficient for each of the independent variable
for idx, col_name in enumerate (X_train.columns):
    print("The coefficient for {} is {}".
          format(col_name,lin_reg.coef_[0][idx]))
```

```
The coefficient for cylinders is -0.6576258418829273
The coefficient for displacement is 2.330251896461423
The coefficient for horsepower is -0.7459683255585576
The coefficient for weight is -5.946105977233697
The coefficient for acceleration is 0.16973210125549099
The coefficient for model year is 2.9241345021479934
The coefficient for origin is 0.9598668889216055
```

In [30]:
```
# Intercept value
intercept = lr.intercept_[0]
print("The intercept for our model is {}".format(intercept))
```

```
The intercept for our model is 23.600719424460433
```

In [31]:
```
y_train_pred = lr.predict(X_train)
```

In [32]:
```
y_test_pred = lr.predict(X_test)
```

In [33]:
```
lr.score(X_train,y_train)
```

Out[33]:
```
0.8081802739111359
```

In [34]:
```
lr.score(X_test,y_test)
```

Out[34]:
```
0.8472274567567306
```

Model Performance Measure

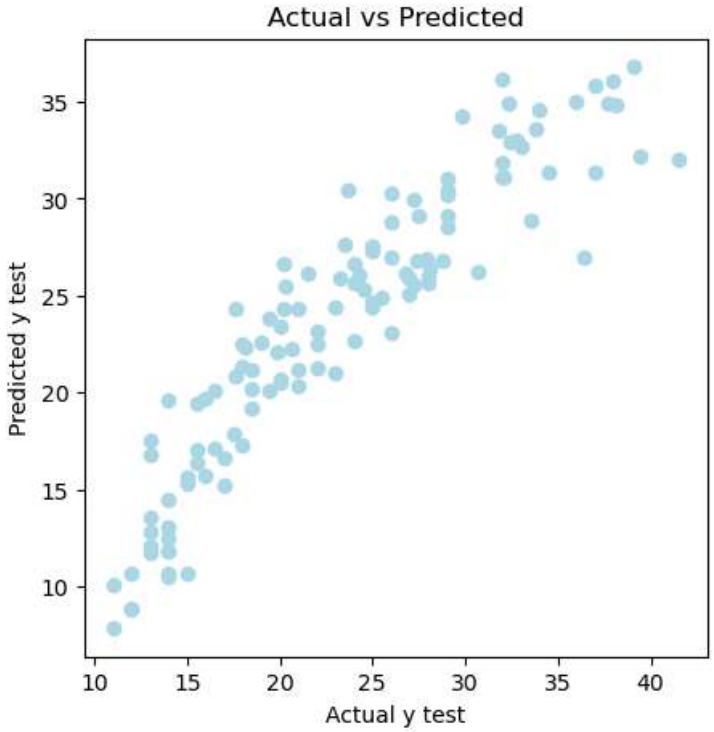
In [35]:
```
#comparing predicted values with actual values of train dataset
from sklearn.metrics import r2_score
r2_score_train = r2_score(y_train,y_train_pred)
r2_score_train*100
```

Out[35]:
```
80.81802739111359
```

In [36]:
```
#comparing predicted values with actual values of test dataset
r2_score_test = r2_score(y_test,y_test_pred)
r2_score_test*100
```

Out[36]:
```
84.72274567567307
```

In [37]:
```
# Visualizing the Actual vs predicted model
plt.figure(figsize=(5, 5))
plt.scatter(x=y_test, y=y_test_pred, color='lightblue')
plt.xlabel('Actual y test')
plt.ylabel('Predicted y test')
plt.title('Actual vs Predicted')
plt.show()
```



Inference:

- The above plot show the linearity between actual and predicted shows the linear regression is successfull.
- Visualizations and plots shows the relationship between the features.
- We checked for duplicates and missing values for Data Cleaning.
- We Noticed the numerical column is in object data type by df.info() function.
- Checked for the unreal value and replaced with null and filled with median value.
- Splitted to train and test split to train and test the data for regression model.
- Scaled the data before fitting so that model would be easy and confident to predict the datas.
- Fitting the train data to model and got the score of 80% confident .
- Proceeded to test and predicted it and showed 84% confident score (R2 score).
- Visualized the y -(actual and predicted)
- Linearity is shown by scartter plot.
- So, This model would predict the MPG of cars with these features present with 80-84% Confidence. *

- Important note:

```
*   May this model be 80% confident by we can further tune and get high R2 score.
*   Follow me on linkdin for future updates : Jeevarathnam R T , URL :
www.linkedin.com/in/jeeva46DA
```